Team PLC

Milestone 4

Test Cases and System Design

Harry Henman, Christopher Riley, and Micah Weaver
10/26/2007

Table of Contents

1. Executive Summary

The purpose of this document is to specify the design of the Programming Language Concepts grading script system. Updated features, use cases, and supplementary specifications have been included from the previous milestones for reference and context when looking at the design of the system and test cases.

This document also specifies the test cases which will be used to evaluate how well the system meets the requirements set up by these Milestone documents. To specify the design of the system, system architecture has been included with details about various modules of the system and how they interact. Based on the system architecture, pseudo code is also written to show how the system should function using its various modules. In addition to the system architecture and pseudo code, the database design is included to show how data about assignments, problems, and test cases will be stored. This document sets up a place where the implementation team can look at the document, and know almost exactly how to implement and test the system.

2. Introduction

There are two main goals of Milestone 4. The first is to determine specific test cases which show clearly whether or not the use cases from Milestone 2 and the functionality requirements of Milestone 3 actually perform their desired functions. These test cases will be used to create a test plan for the final deliverables.

The second goal of Milestone 4 is to establish a design of how the system will be implemented. The system architecture, pseudo code, and data design will be used to form the design plan for the final deliverables. This document is setting the stage for the requirements team to transfer control of the project to the implementation and testing team. With this document, the only requirements left to specify in Milestone 5 are those giving a revised interface design to the prototypes from Milestone 3.

3. Features

This section describes features that will be implemented in the final version of the grading script. The priority indicates how important this feature is to the users. A priority of “Critical” indicates that the feature must be included for the grading script to have any worth. A priority of “Important” indicates that the feature is important, but may have some flexibility. A priority of “Useful” indicates that the feature will be nice to have, but is not crucial to the success of the program.

The effort is an estimation of how much work each feature will take to plan and develop. An effort of low indicates that it will not take much effort to implement the feature because it is a simple feature, or because there is an existing process which can be reused. A task with medium effort may indicate that some research has to be done or some work has to be done to make the existing process fit with the new system. Finally, high effort items will take a significant amount of research time, implementation time, and testing and debugging time.

The risk shows the possibilities that implementing this feature has towards negatively affecting another feature or delaying the development of the project. Low risk indicates that the feature should not interfere with any other features or delay the project, medium risk indicates that the feature might cause some problems with other features or delay the project, and high risk indicates that the feature will probably either cause features to not work because of design changes, or the feature will cause a delay in releasing the project.

The version indicates the version number in which the feature will be released. Version 1.0 will be implemented during the winter quarter, and if there is sufficient time, Version 2.0 will also be implemented.

Finally, the reason describes why this feature should be included in the grading script solution.

#	Feature Name	Priority	Effort	Risk	Version	Reason
1	All users can upload solutions.	Critical	Low	Medium	1.0	If students cannot upload their solutions, there is no way for the grading script to access their work. Professors and teaching assistants need to upload solutions to test the test cases.
2	When solutions are uploaded, they are automatically evaluated.	Important	Low	Low	1.0	When users submit solutions, they should see the results immediately.
3	Professors and teaching assistants can grade all students' work and receive a complete report.	Critical	Medium	Medium	1.0	The professors and teaching assistants need to be able to grade the entire class's work.
4	Professors and teaching assistants can grade a single student's work.	Critical	Low	Medium	1.0	If a student submits late work, it should be graded without grading every student's work.
5	Reports are stored and can be viewed at a later time.	Critical	Medium	Low	1.0	Users should be able to view an existing report without being forced to upload the solution and make the system recreate the report.
6	Each assignment has a configurable time allotment for each test case to run which overwrites a default time allotment. If a student's solution exceeds that time, the grading script assigns zero points to the solution for that test case and continues to the next test case.	Critical	Low	Medium	1.0	If student solutions lead to infinite loops or very inefficient algorithms, the server becomes bogged down trying to compute a never-ending task. This consumes system resources, hampering the ability for other students to use the grading script, and must be terminated.

7	Test cases are not available to students through the execution of malicious Scheme code [1].	Critical	High	High	1.0	The homework is meant to see if the students can apply the concepts, not if they can engineer a program to meet certain test cases.
8	Kerberos [2] identification will be used to authenticate and identify users, and only students, teaching assistants, and professors in the class during the current term have access to the system.	Critical	Medium	Medium	1.0	Some form of user authentication is necessary to ensure that users have the rights they are supposed to have. Using Kerberos [2] will be good, because the students already have their username and password.
9	Professors and teaching assistants can create new assignments with problems, a due date, a time allotment, and loaded modules.	Critical	Medium	Medium	1.0	Students can only submit solutions to an assignment if the assignment actually exists.
10	Professors and teaching assistants can create new problems with test cases and a description.	Critical	Medium	Medium	1.0	Each assignment is broken up into problems.
11	Professors and teaching assistants can create new test cases with the code to run, a comparison operator, and the possible point value.	Critical	Medium	Medium	1.0	The course and assignments will change from time to time, so the professors and teaching assistants must be able to assign new test cases.
12	Professors and teaching assistants can modify and delete existing test cases.	Critical	Medium	Low	1.0	Most of the time, when creating new test cases, mistakes are made. The professors and teaching assistants uploading test cases cannot be expected to produce perfect test cases the first time.
13	Professors and teaching assistants can modify and delete existing problems by adding or removing test cases and changing the problem information.	Critical	Medium	Low	1.0	From quarter to quarter, the professors may change the test cases for each problem to keep new students from getting the test cases from students who have already taken Programming Language Concepts [PLC] [3].

14	Professors and teaching assistants can modify and delete existing assignments by adding or removing problems and changing the assignment information.	Critical	Medium	Low	1.0	Professors may need to add or remove problems as they see fit or even to extend the due date for a particular assignment.
15	Students can view a detailed report with the tests cases and their answers in limited length using Scheme's [1] *pretty-print* [4] function after the assignment due date has passed.	Critical	Medium	Medium	1.0	After the assignment due date, students need to see what they missed to learn from their mistakes and possibly fix their solutions. Limiting the length and using *pretty-print* [4] make the report easy to read.
16	Students can submit solutions as a team.	Critical	High	High	1.0	The last several assignments for PLC [3] are team assignments and should be graded as team assignments.
17	Professors and teaching assistants can copy existing problems into assignments .	Important	Medium	Low	1.0	Professors may move problems around to different assignments over time.
18	Professors and teaching assistants can copy existing test cases into problems .	Important	Medium	Low	1.0	Many of the same test cases are used in several different problems.
19	The time allotment for the assignment can be overwritten by a time allotment for a problem or test case.	Important	Medium	Medium	2.0	Certain problems or test cases may require more time to run than the rest of the assignment.
20	The system collects data each time the grading script is run on scores and submission time.	Important	High	High	2.0	Collecting data on problems graded by the grading script will help the professor know how to improve the class and student learning.
21	The final grade will be recorded and sent to students automatically.	Useful	Medium	Medium	2.0	Having the final grade recorded and sent to the students will be useful, but the students do have other ways to find out their grade.
22	There will be a drop-down menu or text box to select which problems to grade.	Useful	Medium	Medium	2.0	Allowing only specific problems to be specified when grading will eliminate time spent grading problems that the user already knows are correct.

23	Students can drag-and-drop code to upload.	Useful	Medium	Low	2.0	Allowing users to drag and drop code from their operating system environment to the website will save the user from having to browse for specific files.

4. Use Cases

The use cases provide a control flow for the features listed above. They make sure that the features from Version 1.0 are accounted for in the design. For each use case, there is a name, description, list of actors, basic flow, alternative flow, set of pre-conditions, and a set of post-conditions. Some use cases also have special requirements and other stakeholders, and all use cases have a set of features satisfied.

The description briefly describes the use case. The actors are the users or systems that are directly involved in the use case. The basic flow is a layout of what happens during a typical execution of the use case. Alternate flows list things that might occur differently during the process of the basic flow. Pre-conditions and post-conditions detail the state of the system before and after the flow is executed. Special requirements are system requirements that did not fit as part of the flow, pre-conditions, or post-conditions. Other stakeholders are stakeholders that might be indirectly affected by the execution of a use case. Finally, the features satisfied correspond to the numbered features that each use case fulfills.

4.1 Student Submit Solution

Description:	The user submits a solution, the solution is evaluated, and a report is returned to the user and stored on the server.
Actors	Any user
Basic flow:	1. The user selects an assignment for which to upload a solution. 2. The user clicks “Upload solution”. 3. The user is taken to a page to upload the solution for the assignment chosen. 4. The user selects a file or files to upload. 5. The user selects “Submit”. 6. The solution is stored on the server. 7. The grading script grades the user’s solution and assigns scores for each problem using the test case information for that assignment. 8. A partial report is returned to the user with the score earned for each problem and the score possible for each problem, and that report is stored on the server.
Alternate flows:	1. During step four, the user clicks “Cancel” and is returned to the page for that assignment. 2. During step six, an error occurs either connecting to the server or locating the user’s file. The user is notified by an error message, and nothing is stored on the server. After exiting the error message, the user is returned to the assignment page. 3. During step seven, if the solution takes more time to grade for a particular test case than the allotted time, then the score for that test case is zero points, and the next test case is performed. 4. If the student’s solution contains a Scheme [1] tag noting that the project was a group project, the report is stored for each member of the team in step eight. 5. The user clicks “Back” on any page and goes back one page.
Pre-conditions:	• The user has been authenticated and is at the entry page for the system.
Post-conditions:	• Basic flow and alternate flows 3, 4, and 5: The solution and report are stored on the server, and the user is viewing the report. • Alternate flows 1 and 2: The user is on the assignment page and nothing new has been stored on the server.
Special requirements:	• Test cases are not available to the user through the execution of malicious Scheme [1] code.
Features satisfied:	1, 2, 5, 6, 7, and 16

4.2 Professor Grade All Solutions

Description:	Professors or teaching assistants use the system to grade all students' work for a particular assignment.
Actors	Teaching assistants and professors
Basic flow:	1. The user selects the assignment to grade. 2. The system takes the user to the page for that assignment. 3. The user clicks “Grade all solutions.” 4. The system grades all solutions one at a time and assigns scores to each solution using the test case information for that assignment. 5. The system generates a complete professor report from these scores with each student’s points earned and points possible and stores it for the professor to view. 6. The system generates a complete student report for each solution with the test case function, the student’s output, the correct output, and the points earned and points possible, and the system stores each student’s report for only that student and the professor to view. 7. The system takes the user to a page showing the complete professor report.
Alternate flows:	1. The user clicks “Cancel” during step 4 or 5 and is taken back to the page for the assignment. 2. During step 4 or 5, an error occurs connecting to the server, and the user is notified by an error message.
Pre-conditions:	• The user is an authenticated teaching assistant or professor for the class and is at the entry page for the system.
Post-conditions:	• Basic flow: The user is viewing the report for all solutions. • Alternate flow 1: The user is viewing the assignment page. • Alternate flow 2: The user is viewing an error message.
Features satisfied:	3 and 5

4.3 Professor Grade One Solution

Description:	Professors or teaching assistants use the system to grade one student's work for a particular assignment.
Actors	Teaching assistants and professors
Basic flow:	1. The user selects the assignment to grade. 2. The system takes the user to the page for that assignment. 3. The user selects the username of the student to grade from a drop-down box. 4. The user clicks "Grade one solution." 5. The system grades the student's solution and assigns scores to the solution using the test case information for that assignment. 6. The system generates a report from these scores and stores it. 7. The system takes the user to a page showing the generated report.
Alternate flows:	1. The user clicks "Cancel" during step 5 or 6 and is taken back to the page for the assignment. 2. During step 5 or 6, an error occurs connecting to the server, and the user is notified by an error message.
Pre-conditions:	• The user is an authenticated teaching assistant or professor for the class and is at the entry page for the system.
Post-conditions:	• Basic flow: The user is viewing the report for one student's solution. • Alternate flow 1: The user is viewing the assignment page. • Alternate flow 2: The user is viewing an error message.
Features satisfied:	4 and 5

4.4 Log In

Description:	Users must be authenticated and identified before accessing the system.
Actors	All users and Kerberos [2]
Basic flow:	1. The user types in his or her username and password. 2. The system uses Kerberos [2] to determine the identity of the user or to determine if the user has chosen a valid username and password combination. 3. The system checks internal files to make sure that the user is involved in a PLC [3] class for this particular term. 4. The system creates a session for the user. 5. Professors and teaching assistants are taken to the professor entry page, and students are taken to the student entry page.
Alternate flows:	1. During step 2, if the username and password do not match Kerberos [2] data, the user is returned to the login page with an error message informing him or her that the username and password are not valid together. 2. As a result of step 3, if the user is not in PLC [3] this term, he or she is returned to the login page with a different error message. 3. If the system cannot connect to Kerberos [2] during step 2, the user is returned to the login page with an error message.
Pre-conditions:	• The user is at the login page for the system.
Post-conditions:	• Basic flow: The user is at the entry page for his area of the system with a new secure session. • Alternate flows: The user is viewing an error message at the login page.
Special Requirements:	Login credentials must be transmitted securely, and not stored in the system.
Features satisfied:	8

4.5 Create Assignment

Description:	Professors and teaching assistants create new assignments for the course.
Actors	Professors and teaching assistants
Basic flow:	1. The user clicks the "Create new assignment" link. 2. The user is taken to a page with a form for the new assignment. 3. The user fills in the due date, time allotment for code to execute, and loaded modules associated with the assignment. 4. The user selects "Continue". 5. The user adds problems, via the use case "Create Problem", to the assignment. 6. The user clicks "Done", the information is uploaded to the database, and the user is returned to the entry page for the professor portion of the system.
Alternate flows:	1. If the user clicks "Cancel" on the assignment page, he or she is returned to the entry page for the professor portion of the system. 2. During step 3, the due date has passed or the loaded modules are not available, an error is shown and the form is re-displayed so input can be re-entered. 3. During step 6, an error occurs when uploading the problem information, the user is notified by an error message, and nothing is stored on the server.
Pre-conditions:	• The user is an authenticated PLC [3] professor or teaching assistant and is at the entry page for the system.
Post-conditions:	• Basic flow and alternate flow 2: The user is at the entry page for the system, and an assignment has been created. • Alternate flow 1: The user is at the entry page for the system, and an assignment has not been created. • Alternate flow 3: The user is at the entry page for the system viewing an error message, and an assignment has not been created.
Other stakeholders:	Students will be able to see the assignment and upload code for it.
Features satisfied:	6 and 9

4.6 Create Problem

Description:	Professors and teaching assistants create new problems for an assignment.
Actors	Professors and teaching assistants
Basic flow:	1. The user clicks the “Add problem” link. 2. The user is taken to a page for the new problem. 3. The user enters a description of the problem and the time allotment for code to execute. 4. The user adds test cases, via the use case “Create Test Case”, to the problem. 5. The user clicks “Done”, the information is uploaded to the database, and is returned to the assignment page or entry page depending on where he came from.
Alternate flows:	1. The user clicks “Cancel” and is taken back to the page for the assignment or the entry page. 2. During step 5, an error occurs when uploading the problem information, the user is notified by an error message, and nothing is stored on the server.
Pre-conditions:	• The user is an authenticated PLC [3] professor or teaching assistant and is at the assignment page or the entry page.
Post-conditions:	• Basic flow: The user is at the assignment page or entry page, and a new problem has been added. If the user came from the assignment page, the problem has been added to the assignment. • Alternate flow 1: The user is at the assignment page or entry page, and a new problem has not been created. • Alternate flow 2: The user is viewing an error message before being returned to the assignment page or entry page, and a new problem has not been created.
Other stakeholders:	Students will be able to see their score for each problem when running the grading script.
Features satisfied:	10

4.7 Create Test Case

Description:	Teaching assistants and professors create new test cases for the system.
Actors	Teaching assistants and professors
Basic flow:	1. The user clicks the “Add test case” link. 2. The user is taken to a page for the new test case. 3. The user inputs the test code to run, a comparison function, the time allotment for code to execute, and the number of points possible. 4. The user clicks “Submit.” 5. The test case is stored on the server. 6. The user is taken back to the page for the problem or the entry page depending on where he came from
Alternate flows:	1. The user clicks “Cancel” and is taken back to the page for the problem or the entry page. 2. An error occurs when uploading the test case information in step 5, the user is notified by an error message, and nothing is stored on the server.
Pre-conditions:	• The user is an authenticated teaching assistant or professor for the class and is at the entry page for the system.
Post-conditions:	• Basic flow: The user is at the problem page or entry page, and a new test case has been added. If the user came from the problem page, the test case has been added to the problem. • Alternate flow 1: The user is at the problem page or entry page, and a new test case has not been created. • Alternate flow 2: The user is viewing an error message before being returned to the problem page or entry page, and a new test case has not been created.
Other stakeholders:	Student solutions are graded using these test cases.
Feature satisfied:	11

4.8 Edit Test Case

Description:	Teaching assistants and professors edit existing test cases for the system.
Actors	Teaching assistants and professors
Basic flow:	1. The user clicks the "Edit test case" link. 2. The user is taken to the test case information page. 3. The user can edit the test code to run, a comparison function, the time allotment for code to execute, and the number of points possible. 4. The user clicks "Submit." 5. The test case is stored on the server. 6. The user is taken back to the page for the problem.
Alternate flows:	1. The user clicks "Cancel" and is taken back to the page for the problem or the entry page. 2. The user clicks "Delete," the test case is deleted from the system, and the user is taken back to the problem page or entry page. 3. An error occurs when retrieving or rewriting the test case information, the user is notified by an error message, and nothing is stored on the server.
Pre-conditions:	• The user is an authenticated teaching assistant or professor for the class and is at a page for a problem containing the test case or at a page with a list of test cases.
Post-conditions:	• Basic flow: The user is back at the problem page or entry page, and the test case has been edited. • Alternate flow 1: The user is back at the problem page or entry page, and the test case has not been edited. • Alternate flow 2: The user is back at the problem page or entry page, and the test case has been deleted. • Alternate flow 3: The user is viewing an error message before returning to the problem page or entry page, and the test case has not been edited.
Other stakeholders:	Student solutions are graded using these test cases.
Feature satisfied:	12

4.9 Edit Problem

Description:	Professors and teaching assistants edit problems for an assignment.
Actors	Professors and teaching assistants
Basic flow:	1. The user clicks the "Edit problem" link. 2. The user is taken to the problem information page. 3. The user can edit the description of the problem and the time allotment for code to execute. 4. The user can add test cases to the problem via use case 7, edit the test case via use case 8, or remove test cases from the problem. 5. The user clicks "Done" and is returned to the assignment page or entry page depending on where he came from.
Alternate flows:	1. The user clicks "Cancel" and is taken back to the page for the assignment or the entry page. 2. The user clicks "Delete," the problem is deleted from the system, and the user is taken back to the assignment page or entry page. 3. An error occurs when retrieving or rewriting the problem information, the user is notified by an error message, and nothing is stored on the server.
Pre-conditions:	• The user is an authenticated PLC [3] professor or teaching assistant and is at the assignment page or at a page with a list of problems.
Post-conditions:	• Basic flow: The user is back at the assignment page or entry page, and the problem has been edited. • Alternate flow 1: The user is back at the assignment page or entry page, and the problem has not been edited. • Alternate flow 2: The user is back at the assignment page or entry page, and the problem has been deleted. • Alternate flow 3: The user is viewing an error message before returning to the assignment page or entry page, and the problem has not been edited.
Other stakeholders:	Students will be able to see their score for each problem when running the grading script.
Features satisfied:	13

4.10 Edit Assignment

Description:	Professors and teaching assistants edit assignments for the course.
Actors	Professors and teaching assistants
Basic flow:	1. The user clicks the “Edit assignment” link. 2. The user is taken to the assignment information page. 3. The user can edit the due date, time allotment for code to execute, and loaded modules associated with the assignment. 4. The user can add problems to the assignment via use case 6, edit problems via use case 9, or remove problems from the assignment. 5. The user clicks “Done” and is returned to the entry page for the professor portion of the system.
Alternate flows:	1. The user clicks “Cancel” on the assignment page and is returned to the entry page for the professor portion of the system. 2. The user clicks “Delete,” the problem is assignment from the system, and the user is taken back to the entry page. 3. The due date has passed or the loaded modules are not available, an error is shown, and the form is re-displayed so input can be re-entered. 4. An error occurs when retrieving or rewriting the problem information, the user is notified by an error message, and nothing is stored on the server.
Pre-conditions:	• The user is an authenticated PLC [3] professor or teaching assistant and is at the entry page for the system.
Post-conditions:	• Basic flow and alternate flow 3: The user is back at the entry page, and the assignment has been edited. • Alternate flow 1: The user is back at the entry page, and the assignment has not been edited. • Alternate flow 2: The user is back at the entry page, and the assignment has been deleted. • Alternate flow 4: The user is viewing an error message before returning to the entry page, and the assignment has not been edited.
Other stakeholders:	Students will be able to see the assignment and upload code for it.
Features satisfied:	6 and 14

4.11 View Report

Description:	Users can view reports on performance for a particular assignment.
Actors	All users
Basic flow:	1. The user clicks the “View Report” link. 2. The user is taken to a page displaying the report’s information which is either a professor report, full student report, or partial student report. 3. The user clicks “Done” and is returned to the assignment page.
Alternate flows:	1. There is no existing report for this user to view, and the user is notified by an error message. 2. An error occurs when uploading the report, and the user is notified by an error message.
Pre-conditions:	• The user has been authenticated and is at an assignment page.
Post-conditions:	• Basic flow: The user is at the assignment page and has viewed the report. • Alternate flows: The user is viewing an error message before being returned to the assignment page.
Special Requirements:	The report for a particular student should only be visible to that student, the PLC [3] professor for that term, and the teaching assistants for that term.
Features satisfied:	5, 15

4.12 Copy Problem

Description:	Professors and teaching assistants copy an existing problem into an assignment.
Actors	Professors and teaching assistants
Basic flow:	1. The user clicks the "Copy Existing Problem" link. 2. The user is taken to a page listing all problems made to date. 3. The user clicks on a particular problem. 4. The system adds the selected problem to the assignment in progress. 5. The user is taken back to the page for the assignment, and the problem is shown among the list of problems that compose the assignment.
Alternate flows:	1. The user hits the back button and returns to the page for the assignment. 2. An error occurs when retrieving or rewriting the problem information, the user is notified by an error message, and nothing is stored on the server.
Pre-conditions:	• The user is an authenticated PLC [3] professor or teaching assistant and is at the assignment page.
Post-conditions:	• Basic flow: The user is back at the assignment page, and the problem has been added to the assignment. • Alternate flow 1: The user is back at the assignment page, and the problem has not been added to the assignment. • Alternate flow 2: The user is viewing an error message before returning to the assignment page, and the problem has not been added to the assignment.
Other stakeholders:	Students will be able to see their score for each problem when running the grading script.
Features satisfied:	17

4.13 Copy Test Case

Description:	Professors and teaching assistants copy an existing test case into a problem.
Actors	Professors and teaching assistants
Basic flow:	1. The user clicks the "Copy Existing Test Case" link. 2. The user is taken to a page listing all test cases made to date. 3. The user clicks on a particular test case. 4. The system adds the selected test case to the problem in progress. 5. The user is taken back to the page for the problem, and the test case is shown among the list of test cases that compose the problem.
Alternate flows:	1. The user hits the back button and returns to the page for the problem. 2. An error occurs when retrieving or rewriting the test case information, the user is notified by an error message, and nothing is stored on the server.
Pre-conditions:	• The user is an authenticated PLC [3] professor or teaching assistant and is at the problem page.
Post-conditions:	• Basic flow: The user is back at the problem page, and the test case has been added to the problem. • Alternate flow 1: The user is back at the problem page, and the test case has not been added to the problem. • Alternate flow 2: The user is viewing an error message before returning to the problem page, and the test case has not been added to the problem.
Other stakeholders:	Student solutions are graded using these test cases.
Features satisfied:	18

5. Supplementary Specification

The purpose of this section of the document is to collect and organize those requirements not currently captured in the use cases developed for the system. These requirements fall into several distinct categories including functionality, usability, reliability, performance, and others. Many requirements have an associated use case which is indicated by the number preceding its description. These requirements are based off of the proposed list of features for release one.

5.1 Functionality

- If a student uploads code while the program is grading, then the current code must finish executing before the next assignment is graded.
- When an assignment is due, the program will grade each student's submission and produce a final report for every student, plus one final report for the professor with each student's score.
- Grade reports for students will include a list of the problems, the score received for each problem, and the maximum score possible. They will also include the total score and the total score possible.
- The professor's grade report will include the following information for each student sorted alphabetically by the student usernames: name of problem, points earned for each test case, points possible for each test case, total points earned and total points possible.
- When a student's code is evaluated for a particular test case, if output is equivalent to the correct value, based on the given comparison function, then all of the points possible for that test case are awarded to the student.

5.2 Usability

- Regular users will be able to upload their solutions within one minute of logging in.
- At least 90% of new users should find uploading solutions similar to the method of uploading files used by ANGEL [5].
- Authorized users should be able to grade all solutions or grade one solution within twenty seconds of logging in.
- Users should be able to log in within ten seconds of coming to the log in page assuming that they already have usernames and passwords.
- Users should find logging in to this system similar to logging into ANGEL.
- Authorized users should be able to create already prepared test cases in less than one minute on average once they are logged in.
- Authorized users should be able to create new test cases, problems, and assignments without using the help sections after one use on average.
- Authorized users should be able to edit test cases, problems, and assignments without using the help sections after one use on average.
- Users should be able to perform any task without referring to the help section after using the help section to perform the task the first time.
- Every screen must have a link to an associated help section.

5.3 Reliability

- The features of the system connected to submitting solutions and viewing reports must be available 99 percent of the time with no difference between business and non-business hours.
- The security of test cases must be maintained at all times.
- The mean time between failures is one week where a failure is any malfunction of the system which prevents users from submitting solutions and receiving reports.
- The mean time to repair failures which prevent users from submitting solutions and receiving reports is one hour assuming that someone is able to work on the repair at the moment the failure occurs.
- Users observe less than one error in every one thousand attempts when uploading solutions, excluding errors that occur because the user's file is not in the directory he or she specified.
- There is only one error in every fifty attempts when generating reports.
- The grade for students will not be computed as a percent; it will remain as an unreduced fraction of points earned over points possible.
- Unauthenticated users will never be allowed to use the system.

5.4 Performance

- Once a solution has been submitted, the evaluation requires one second on average plus the amount of time needed to evaluate the same test cases on the student's machine.
- The system can accommodate twenty users at a time.
- If there are more than twenty users, the system continues to operate, but may be noticeably slower.

5.5 Supportability

- The web interface should work with all common browsers, most notably Internet Explorer [6], Firefox [7], Opera [8], and Safari [9].
- Small modifications should be easily completed within one business day.
- Medium modifications should be completed within one week.
- Large modifications such as the release of a new version of Chez Scheme [3] should be completed within two weeks.

5.6 Design Constraints

- If an entirely new grading script is developed, then it should be written in Chez Scheme [3] to use the existing knowledge of Dr. Anderson.
- The system must be designed to accommodate a change in the programming language in which student solutions are implemented.
- The system must be able to be run on a UNIX [10] server.

5.7 User Documentation and Help System Requirements

- There must be a link on every page of the system to a user's manual explaining the basic functionality of the system and how to use it.

5.8 Interfaces

5.8.1 User Interfaces

- All users will interact with the system through a web page. Students will have a limited set of options while professors will have a larger variety of ways to interact with the system. For more detail, see the user interface prototypes from Milestone 3.

5.8.2 Software interfaces

- The system must interface with a database containing assignments, problems, and test cases. The database is a separate module of the system.
- The system must be able to store user solutions on the server as separate files.
- The system must be able to read user solutions from the server.
- The system must be able to store reports on the server as separate files.
- The system must be able to read reports from the server.
- The system will also interact with Kerberos [2] for obtaining authentication information.

5.9 Licensing and Security Requirements

- This system may only be used at Rose-Hulman Institute of Technology [11].
- The system may not be sold to parties outside of Rose-Hulman Institute of Technology [11] without the consent of the requirements and design team and the implementation and testing team.

5.10 Legal, Copyright, and Other Notices

- This system comes as is with no warranty.
- The intellectual property behind the system belongs to the requirements and design team and the implementation and testing team.

5.11 Internationalization and Localization

- This solution only needs to be usable by students who can understand English.
- The solution could be expanded to include foreign languages in a later release.

5.12 Physical Deliverables

- All requirements documents will be delivered to Dr. Anderson, the client prior to the release of the system.

5.13 Installation and Deployment

- The system will be installed and tested on the Computer Science and Software Engineering department's server by the implementation and testing team before the end of the winter quarter in 2008.

6. Test Cases

The following are the test cases that will be used to evaluate the correctness of the implementation of the system. Each test case has an associated use case, the scenario for each use case, a description, several conditions which determine the flow of the use case, and the expected result of executing the test case.

Use Case	Scenario	Description	User Authenticated	Conditions	Conditions	Expected Result
1	1	A student submits a solution.	True	The assignment has been created.	All problems in the assignment have test cases.	A report screen appears with the correct grade for each problem in the assignment
1	2	The user presses cancel before submitting a solution.	True	The cancel button has been pushed		User is returned to the assignment page.
1	3	An error occurs while connecting to the server during submission.	True	The user's machine cannot connect to the server		An error message appears and the user is directed back to the assignment page. Nothing is stored on the server.
1	4	A solution takes too long to grade upon submission.	True	The time allotted for the evaluation of a particular problem is exceeded.		A score of zero is recorded for that problem and the remaining test cases are graded.

1	5	A student indicates that they are part of a team for this assignment.	True	A tag is present in the student submission noting that the student was part of a group		A grade report is stored for each member of the indicated team.
1	6	The user presses the back button in their browser.	True	The back button is pressed		The user is brought back one page.
2	1	The professor selects the option to grade all submitted student assignments.	True	The assignment has been created.	All problems in the assignment have test cases.	A report screen appears with an abbreviated display of each student's correct grade on every problem.
2	2	The user presses cancel before attempting to grade submissions.	True	The cancel button has been pushed		User is returned to the assignment page.
2	3	An error occurs while connecting to the server.	True	The user's machine cannot connect to the server		An error message appears and the user is directed back to the assignment page.
3	1	The professor selects the option to grade one particular student's assignment.	True	The student has submitted a solution to the assignment	All problems in the assignment have test cases	A detailed report screen appears with the correct grade for every problem.
3	2	The user presses cancel before grading a solution.	True	The cancel button has been pushed		User is returned to the assignment page.
3	3	An error occurs while connecting to the server.	True	The user's machine cannot connect to the server		An error message appears and the user is directed back to the assignment page.

4	1	A user attempts to log in to the system.	False	User inputs a proper username and password		Professors are directed to their main page and students are directed to a different main page. Both are identified as authenticated for the current session.
4	2	A user attempts to log in to the system with an improper username or password.	False	User inputs a password or username which does not match Kerberos data.		The user is returned to the login page and receives an error message.
4	3	A user is no longer a current PLC student.	False	User inputs a correct Kerberos ID.	User is not enrolled in PLC as either a student, professor or teaching assistant	The user is returned to the login page and receives an error message indicating why they were not allowed to log in.
4	4	There is a problem connecting to the Kerberos system.	False	There is an error in communicating to Kerberos.		The user is returned to the login page and is notified that the system was unable to communicate with Kerberos.
5	1	A user creates a new assignment.	True			A new link appears on both student and professor main pages, and the corresponding assignment is added to the database.
5	2	The user presses cancel on the assignment page.	True	The cancel button has been pushed		User is returned to the entry page.
5	3	The assignment has been created with invalid data.	True	Either a due date is already passed, or a loaded module cannot be found.		An error message is shown and the form is redisplayed with all current data intact.
5	4	An error occurs while connecting to the server.	True	The user's machine cannot connect to the server		An error message appears and the user is directed back to the assignment page.

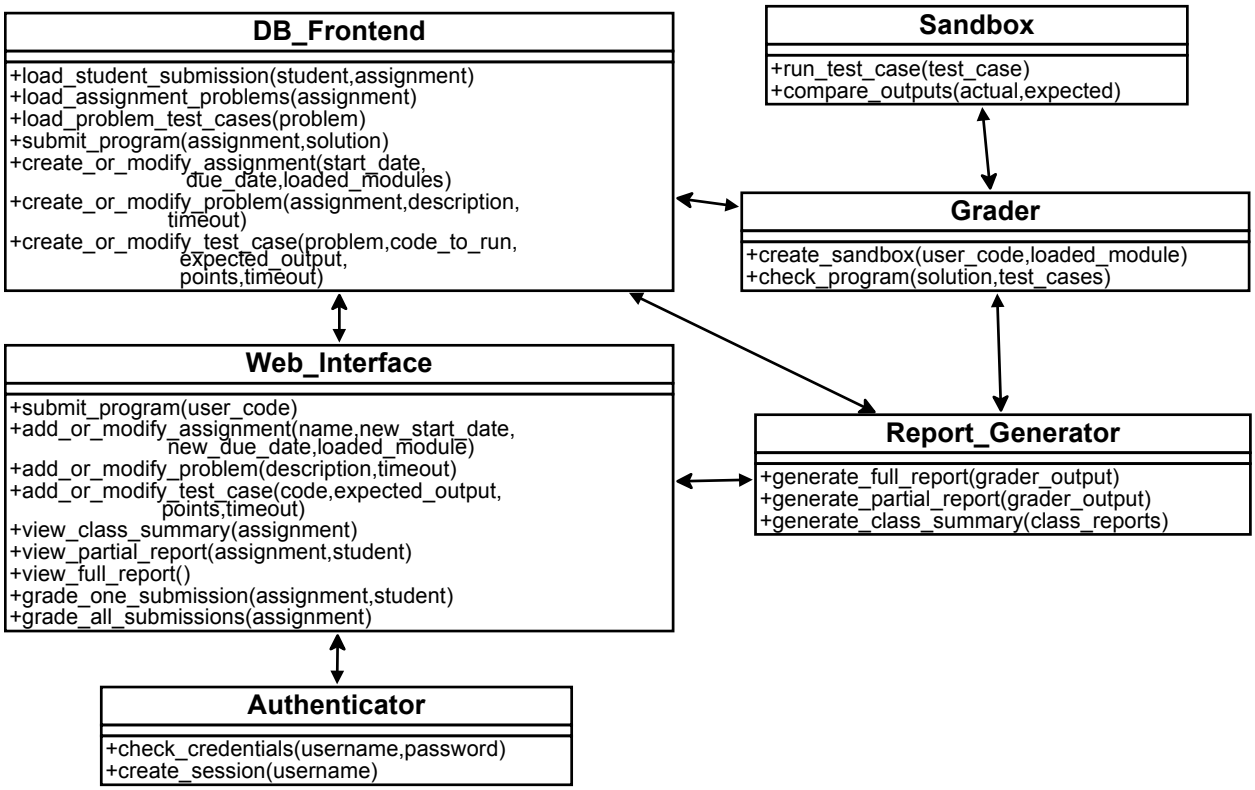
6	1	A user creates new problems and adds them to an existing assignment.	True	An assignment exists which could receive new problems	A new link appears on the professor's assignment page, and the corresponding problem is added to the database.
6	2	The user presses cancel before submitting a problem.	True	The cancel button has been pushed	User is returned to the assignment or entry page.
6	3	An error occurs while connecting to the server.	True	The user's machine cannot connect to the server	An error message appears and nothing is stored on the server.
7	1	A user creates a new test case for use with a particular problem.	True	A problem exists which could have test cases added to it.	A new test case appears on the problem page and the corresponding information is added to the database.
7	2	The user presses cancel before submitting a test case.	True	The cancel button has been pushed	User is returned to the problem or entry page.
7	3	An error occurs while connecting to the server.	True	The user's machine cannot connect to the server	An error message appears and nothing is stored on the server.
8	1	A user edits existing test cases which are associated with a particular problem.	True	Test cases exist for the particular problem.	The test case is updated on the problem page as well as in the database, per the edits made.
8	2	The user presses cancel before submitting a change to a test case.	True	The cancel button has been pushed	User is returned to the problem or entry page.
8	3	The user deletes the current test case.	True	The delete button has been pushed.	The user is returned to the problem or entry page, and the test case being edited is deleted from the system.
8	4	An error occurs while connecting to the server.	True	The user's machine cannot connect to the server	An error message appears and nothing is stored on the server.
9	1	A user edits the description of a problem.	True	A problem exists that can be edited.	The problem's information is updated on the problem page as well as in the database.

9	2	The user presses cancel before submitting a change to a problem.	True	The cancel button has been pushed	User is returned to the assignment or entry page.
9	3	The user deletes the current problem.	True	The delete button has been pushed.	The user is returned to the assignment or entry page, and the problem being edited is deleted from the system.
9	4	An error occurs while connecting to the server.	True	The user's machine cannot connect to the server	An error message appears and nothing is stored on the server.
10	1	A user edits the attributes of a particular assignment.	True	An assignment exists that can be edited.	The main pages and assignment pages are updates for professors as well as for students. The change is made in the database as well.
10	2	The user presses cancel before submitting a change to an assignment.	True	The cancel button has been pushed	User is returned to the entry page.
10	3	The user deletes the current assignment.	True	The delete button has been pushed.	The user is returned to the entry page, and the assignment being edited is deleted from the system.
10	4	An invalid input is received	True	The due date has passed or a loaded module cannot be found.	An error message appears and the form is displayed again with all of the current data.
10	5	An error occurs while connecting to the server.	True	The user's machine cannot connect to the server	An error message appears and nothing is stored on the server.
11	1	A user views the most recent report for a chosen assignment.	True	The chosen assignment has an associated report for the student's work.	The report page opens; it contains all of the data on the originally generated report.

11	2	A user attempts to view a report when none are available.	True	The chosen assignment does not have an associated report for the student's work.	The user is notified by an error message.
11	3	An error occurs while connecting to the server.	True	The user's machine cannot connect to the server	An error message appears.
12	1	A user copies a problem to an assignment	True	A problem exists that can be copied.	The problem is copied onto the assignment page as well as in the database.
12	2	The user presses "Back" before copying a problem.	True	The "Back" button has been pushed	User is returned to the assignment page.
12	3	An error occurs while connecting to the server.	True	The user's machine cannot connect to the server	An error message appears and nothing is stored on the server.
13	1	A user copies a test case to a problem	True	A test case exists that can be copied.	The test case is copied onto the problem page as well as in the database.
13	2	The user presses "Back" before copying a test case.	True	The "Back" button has been pushed	User is returned to the problem page.
13	3	An error occurs while connecting to the server.	True	The user's machine cannot connect to the server	An error message appears and nothing is stored on the server.

7. System Architecture

The system architecture details the various modules of the system along with how the modules work together. Each module has various functions along with the use cases that those functions satisfy. The details of each module follow the diagram.



7.1 DB_Frontend

The Database Frontend Module is responsible for storing and accessing system data. This includes test cases, problems, assignments, submitted solutions, and grade reports. Most of the other modules query it to load, change, or add data.

7.1.1 load_student_submission

The function *load_student_submission* allows other parts of the system (most notably the Grader module) to load a student's solution.

Related use cases: Submit Solution, Grade All Solutions, Grade One Solution

7.1.2 load_assignment_problems

The function *load_assignment_problems* loads the problems for a given assignment. The Grader module needs the problems so it can get to the test cases. Additionally, the interface loads the problems to be edited when a professor edits an assignment.

Related use cases: Submit Solution, Grade All Solutions, Grade One Solution, Edit Assignment

7.1.3 load_problem_test_cases

The function *load_problem_test_cases* loads the test cases for a given problem. The Grader module uses this function to get to the test cases. Additionally, the interface loads the test cases to be edited when a professor edits a problem.

Related use cases: Submit Solution, Grade All Solutions, Grade One Solution, Edit Problem

7.1.4 submit_program

The function *submit_program* allows students to submit their solutions to the system.

Related use cases: Submit Solution, Grade All Solutions, Grade One Solution, Edit Problem

7.1.5 create_or_modify_assignment

The function *create_or_modify_assignment* creates a new assignment (with no problems) in the system, or modifies an existing one.

Related use cases: Create Assignment, Edit Assignment

7.1.6 create_or_modify_problem

The function *create_or_modify_problem* creates a new problem (and adds it to the assignment) or modifies an existing one.

Related use cases: Create Problem, Edit Problem

7.1.7 create_or_modify_test_case

The function *create_or_modify_test_case* creates a new test case (and adds it to the problem) or modifies an existing one.

Related use cases: Create Test Case, Edit Test Case

7.2 Web Interface

The Web Interface Module represents the portion of the system responsible for taking in the input given by the user through a web browser. This includes all possible functionality a user can initiate. Every function corresponds to a screen on the user interface. Essentially, this module drives the other modules.

6.2.1 submit_program

The function *submit_program* allows students to submit solutions.

Related use cases: Submit Solution

7.2.2 add_or_modify_assignment

The function *add_or_modify_assignment* modifies an assignment or creates a new one.

Related use cases: Create Assignment, Edit Assignment

7.2.3 add_or_modify_problem

The function *add_or_modify_problem* modifies a problem or creates a new one. The timeout is the default timeout for test cases for the given problem.

Related use cases: Create Problem, Edit Problem

7.2.4 add_or_modify_test_case

The function *add_or_modify_test_case* modifies a test case or creates a new one. The timeout specifies how long the test runs before a zero grade is assigned.

Related use cases: Create Test Case, Edit Test Case

7.2.5 view_class_summary

The function *view_class_summary* allows professors to view all grades students received on a given assignment.

Related use cases: Grade All Solutions

7.2.6 view_partial_report

The function *view_partial_report* allows students to view their solution's report before the assignment is due. The report is only a partial report before the assignment is due, so only the points scored are shown.

Related use cases: Submit Solution, View Report

7.2.7 view_full_report

The function *view_full_report* allows professors and students to view complete reports after the due date has passed. It shows which test cases passed and failed with the solution's outputs and the correct outputs.

Related use cases: Grade One Solution, View Report

7.2.8 grade_one_submission

The function *grade_one_submission* allows a professor to grade a particular student's submission. This is primarily used when the student's solution is corrected and is being re-graded.

Related use cases: Grade One Solution

7.2.9 grade_all_submissions

The function *grade_all_submissions* grades all student solutions for the given assignment.

Related use cases: Grade All Solutions

7.3 Authenticator

The Authenticator module is responsible for communicating with the Kerberos system to determine if supplied credentials are correct, and to check to make sure that the user is a current PLC professor, teaching assistant, or student.

7.3.1 check_credentials

The function *check_credentials* communicates with the Kerberos system to check validity, and makes sure the user is in the PLC section as well.

Related use cases: Log In

7.3.2 create_session

The function *create_session* creates an authenticated session for the web interface to use.

Related use cases: Log In

7.4 Report_Generator

The Report Generator module is responsible for generating the reports from the output of the Grader module. It converts them to Hypertext Markup Language (HTML) and stores them using the DB_Frontend module so that users can access them.

7.4.1 generate_full_report

The function *generate_full_report* generates a detailed report, showing test-case by test-case output.

Related use cases: Grade All Solutions, Grade One Solution, View Report

7.4.2 generate_partial_report

The function *generate_partial_report* generates a partial report, showing only points awarded.

Related use cases: Submit Solution, View Report

7.4.3 generate_class_summary

The function *generate_class_summary* generates an HTML page summarizing the entire class's grades for an assignment.

Related use cases: Grade All Solutions

7.5 Grader

The Grader module is responsible for taking in the test cases, student code, and assignment parameters from the DB_Frontend module and then grading the code. It sends output about which test cases passed and failed to the Report_Generator module so that users can view HTML reports instead of having to interpret textual output.

7.5.1 create_sandbox

The function *create_sandbox* communicates with the sandbox module, creating an environment in which user code can be run.

Related use cases: Submit Solution, Grade All Solutions, Grade One Solution

7.5.2 check_program

The function *check_program* grades the student's solution, developing output that can then be sent to the report generator.

Related use cases: Submit Solution, Grade All Solutions, Grade One Solution

7.6 Sandbox

The Sandbox module is responsible for creating a sandbox, which is an environment where user can be run with restrictions on the code that they can run. In this case, the restriction is that the student's code should not help the student determine the actual content of the test cases.

7.6.1 run_test_case

The function *run_test_case* executes the test case in the safe sandbox environment.

Related use cases: Submit Solution, Grade All Solutions, Grade One Solution

7.6.2 compare_outputs

The function *compare_outputs* compares the correct test case output to the student's output using the specified comparison operator.

Related use cases: Submit Solution, Grade All Solutions, Grade One Solution

8. Pseudo Code

This section contains pseudo code which provides an overview of how each function in the system architecture works.

8.1 DB_Frontend

```
load_student_submission (student, assignment)
  ask database for the code for assignment written by student

load_assignment_problems (assignment)
  ask database for the problems from assignment

load_problem_test_cases (problem)
  ask database for the test cases for problem

submit_program (student, assignment, solution)
  insert into database solution, for assignment by student

create_or_modify_assignment (assignment_name, start_date,
                                    due_date, loaded_modules)
  if (assignment_name in database)
    '()
    (create new assignment in table)
    set start field to start_date
    set due field to due_date
    set modules field to loaded_modules
```

```
create_or_modify_problem (assignment, name, description, timeout)
  if (name in (load_assignment_problems assignment))
    '()
    (create new problem with name, name)
    assign the new problem to assignment
    set description field to description
    set timeout field to timeout

create_or_modify_test_case (problem, name, code_to_run,
                            expected_output, points, timeout)
  if (name in (load_problem_test_cases problem))
    '()
    (create new test case for problem with name, name)
    set code field to code_to_run
    set answer field to expected_output
    set points field to points
    set timeout field to timeout

store_report (report)
  add report to database, tagged with data from the report

load_report (username, assignment)
  request report for username's assignment from database
```

8.2 Web Interface

```
submit_program (user_code)
  assignment = current-assignment
  (DB_Frontend 'submit_program username, assignment, user_code)

add_or_modify_assignment (name, new_start_date, new_due_date,
                          loaded_module)
    (DB_Frontend 'create_or_modify_assignment
        name new_start_date new_due_date loaded_module)

add_or_modify_problem (description, timeout)
  assignment = current-assignment
  if (create-button pressed)
    (get name for problem)
    (name = current-problem)
  DB_Frontend 'create_or_modify_problem assignment name
                                    description timeout
```

```
add_or_modify_test_case (code, expected_output, points, timeout)
  problem = current-problem
  if (create-button pressed)
    (get name for testcase)
    (name = current-testcase)
  DB_Frontend 'create_or_modify_test_case problem name code_to_run
                                  expected_output points timeout

view_class_summary (assignment)
  (map (lambda (student)
         (DB_Frontend 'view_partial_report assignment student))
         usernames)

view_partial_report (assignment, student)
  full-report = (DB_Frontend 'load_report student assignment)
  display (full-report-scores full-report)

view_full_report (assignment, student)
  (DB_Frontend 'load_report student assignment)

grade_one_submission (assignment, student)
  code = (DB_Frontend 'load_student_submission student assignment)
  problems = (DB_Frontend 'load_assignment_problems assignment)
  test-cases = (map (lambda (problem)
                      (DB_Frontend 'load_problem_test_cases problem)))
  (check-program code test_cases)

grade_all_submissions (assignment)
  (map (lambda (student)
               (grade_one_submission assignment student))
               usernames)
```

8.3 Authenticator

```
check_credentials (username, password)
  pass username and password to Kerberos
  if (Kerberos responds positively)
     (create_session username)
     display error message

create_session (username)
  (make-DB_Frontend (connection info))
  (make-Grader)
  (make-Web_Interface)
  (make-Report_Generator)
```

8.4 Sandbox

```
run_test_case (test_case)
  (let ((engine (make-engine (lambda ()
       (eval (test_case-code_to_run test_case)
                 sandbox-environment))))) ;change the environment
                                          ;while running student code
  (engine (test_case-timeout test_case) ;use the timeout
                 (lambda (x) x)  ;if successful return the answer
                 (lambda (x) 'timeout))) ;if not, then return 'timeout


compare_outputs (actual expected)
  (equal? actual expected)
```

8.5 Grader

```
create_sandbox (user_code, loaded_module)
  (make-sandbox user_code loaded_module (copy-environment
                                         (scheme-environment)))

check-program (solution, test_cases)
  (map (lambda (test_case)
    if (Sandbox 'compare_outputs
               (Sandbox 'run_test_case test_case)
               (test_case-expected_output test_case))
    (test_case-score test_case)  ;the answer is right, full credit
    0)                           ;the answer is wrong, no partial credit
  test-cases) ;do that for every one of these, now we have list of scores
```

8.6 Report_Generator

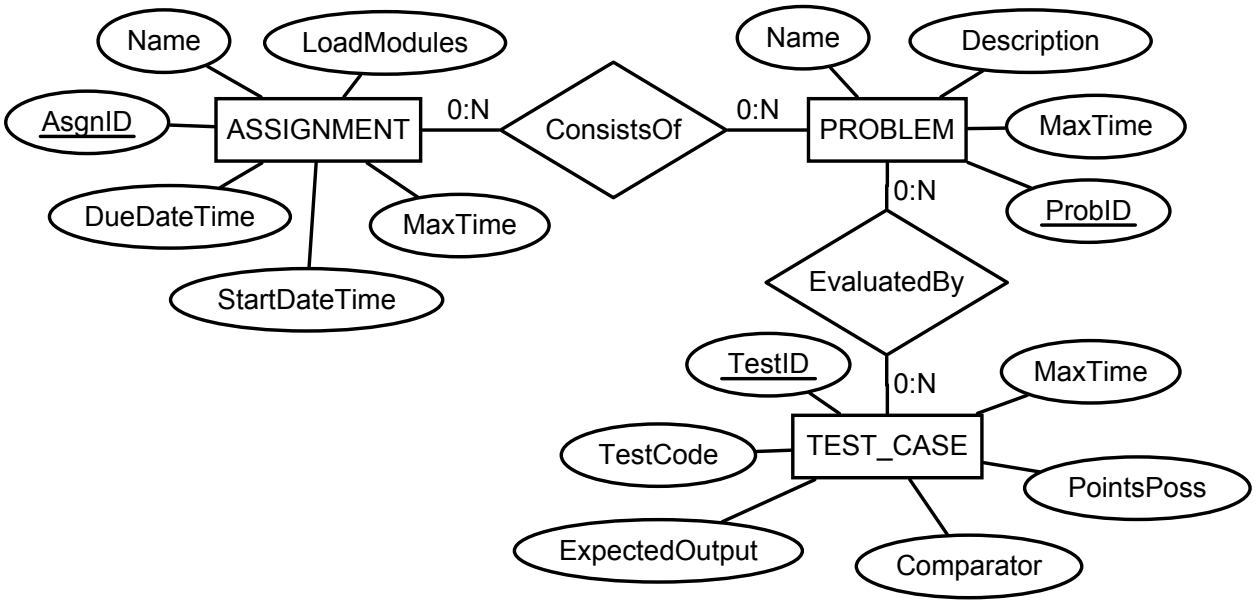
```
generate_full_report (grader_output)
  (display grader_output)

generate_partial_report (grader_output)
  (display (grader_output-scores))

generate_class_summary (class_reports)
  (map display class_reports)
```

9. Data Design

The grading system will store information in two main formats. It will store reports and student solutions on the server as separate files, and it will store test case, problem, and assignment information in a relational database. This section discusses the layout of that relational database.

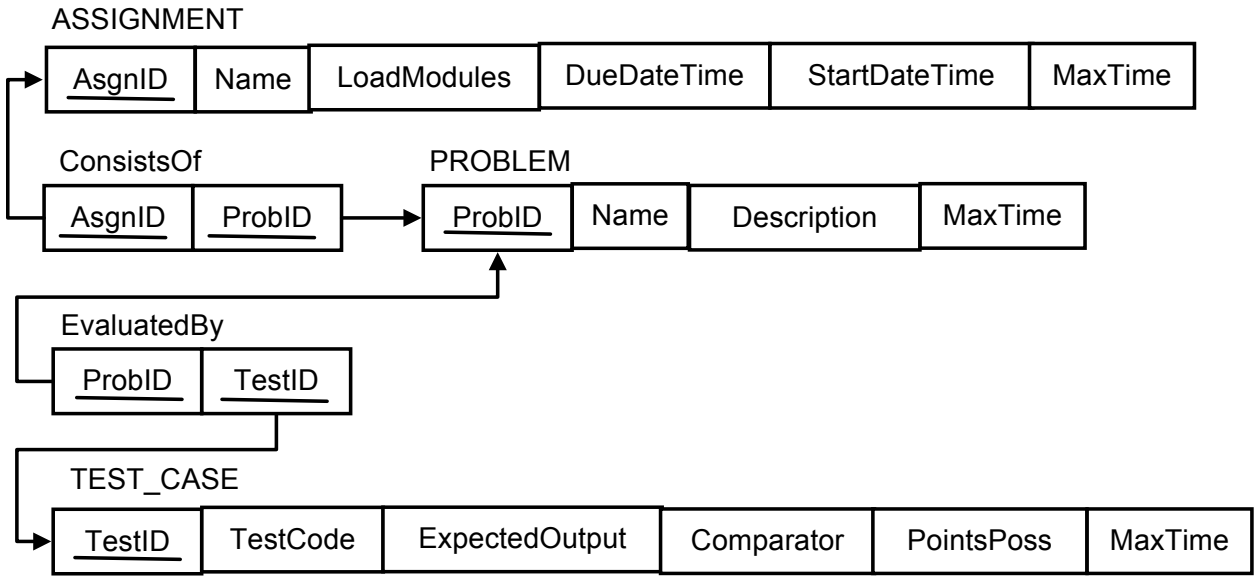
9.1 Entity-Relationship Diagram

The database storage of the assignments, problems, and test cases referred to by the use cases is best described using an entity-relationship diagram. Interpreted, there are three main entities in the database, assignments, problems, and test cases. One assignment consists of any number of problems, while each problem can be evaluated by any number of test cases. All entities have an automatically generated identification number. Assignments have a unique name, a due date and time, a maximum time allotment, and loaded modules. Problems have a unique name, a description, and a maximum time allotment. Test cases have the test code, a comparison function, the expected output, the number of points possible, and a maximum time allotment. Together, these entities, relationships, and attributes define the database backend necessary to support the functionality defined by the use cases.



9.2 Relational Schema

The relational schema translates the entity-relationship diagram into a layout for the database tables and some basic constraints. Each set of boxes represents a table with the table name given above each set. Each box represents a column, and those boxes that are underlined represent keys. Secondary keys have an arrow going from the secondary key to the primary key. All keys with arrows going to them or no arrows represent primary keys.



References

[1] “The Scheme Programming Language.” Chris Hanson, Massachusetts Institute of Technology, 23 October 2003, http://www.swiss.csail.mit.edu/projects/scheme/.

[2] “Kerberos: The Network Authentication Protocol.” Massachusetts Institute of Technology, 5 October 2007, http://web.mit.edu/Kerberos/.

[3] “CSSE 304 Syllabus and Policies.” Department of Computer Science and Software Engineering, Rose-Hulman Institute of Technology, Spring 2007, http://www.rose-hulman.edu/class/csse/csse304/syllabus.html.

[4] “Guile Reference Manual.” The Free Software Foundation, June 2003, http://www.delorie.com/gnu/docs/guile/guile_501.html.

[5] “ANGEL Learning.” ANGEL Learning, 2007, http://www.angellearning.com/.

[6] “Internet Explorer: Home page.” Microsoft, 2007, http://www.microsoft.com/windows/products/winfamily/ie/default.mspx

[7] “Mozilla | Firefox web browser & Thunderbird email client.” Mozilla, 2007, http://www.mozilla.com/en-US/

[8] “Opera browser: Home page.” Opera Software, http://www.opera.com/

[9] “Apple – Safari 3 Public Beta.” Apple Inc., 2007, http://www.apple.com/safari/

[10] “What is UNIX?” The Open Group, 28 December 2003, http://www.unix.org/what_is_unix.html.

[11] “Welcome to Rose-Hulman Institute of Technology.” Rose-Hulman Institute of Technology, 2007, http://www.rose-hulman.edu/

[12] “(CHEZ (CHEZ SCHEME)).” Cadence Research Systems, 2004, http://www.scheme.com/

[13] “RHIT Computer Science and Software Engineering Department.” Department of Computer Science and Software Engineering, Rose-Hulman Institute of Technology, http://www.cs.rose-hulman.edu/.

[14] “HTML 4.01 Specification.” World Wide Web Consortium, 1999, http://www.w3.org/TR/REC-html40/.

[15] “SSH Communications Security.” SSH Communications Security, 2007, http://www.ssh.com/.

Appendix: Traceability Matrix

The traceability matrix shows how the use cases satisfy specific features. Each use case represents a column, and each feature that will be implemented in Version 1.0 represents a row. Each use case must be able to be traced back to a feature, or else it is excessive. Each feature must be able to be traced to a use case to ensure that it is implemented.

	Use Case 1	Use Case 2	Use Case 3	Use Case 4	Use Case 5	Use Case 6	Use Case 7	Use Case 8	Use Case 9	Use Case 10	Use Case 11	Use Case 12	Use Case 13
Feature 1	X												
Feature 2	X												
Feature 3		X											
Feature 4			X										
Feature 5	X	X	X								X		
Feature 6	X				X					X			
Feature 7	X												
Feature 8				X									
Feature 9					X								
Feature 10						X							
Feature 11							X						
Feature 12								X					
Feature 13									X				
Feature 14										X			
Feature 15											X		
Feature 16	X												
Feature 17												X	
Feature 18													X

Index

Glossary

Chez Scheme is an implementation of the Scheme [1] programming language (see Scheme, programming language) developed and maintained by Cadence Research Systems [12]. Chez Scheme is used to execute the Scheme programs that students will submit to the grading script.

Computer Science and Software Engineering (CSSE) is the major used to describe students who study the development of applications for computing systems and how to make these applications better. It is often used to describe the department at Rose-Hulman which teaches these skills, the CSSE [13] department.

Hypertext Markup Language (HTML) [14] provides a means to describe the structure of text-based information in a document – by denoting certain text as headings, paragraphs, lists, and so on – and to supplement that text with interactive forms, embedded images and other objects.

Kerberos [2] is an authentication protocol designed to allow safe access to shared network resources. Kerberos [2] is used at Rose-Hulman Institute of Technology to authenticate students, faculty, and staff prior to granting them access to network-based services.

Pretty-print [4] is a procedure in Scheme which formats nicely and prints an object in Scheme. It is especially useful for displaying large objects in Scheme.

Programming languages are sets of rules for describing a program. They typically use plain, human-readable text to describe the operations a program will perform.

Programming Language Concepts (PLC) is a junior-level course taught to computer science students at Rose-Hulman Institute of Technology. This class teaches advanced computer science topics using the Scheme [1] language (see Scheme)

Scheme [1] is a functional programming language (see programming language) based on the Lisp language. Students in PLC [3] (see PLC) are taught Scheme [1] and then taught advanced programming topics using Scheme [1].

Secure Shell (SSH) [15] is a protocol for executing commands on a remote system securely - that is, without intervening parties being able to deduce the commands run or the output they result in. SSH is also used as a verb, in which case it means to use a program (called an SSH client) to connect to a server that supports the SSH protocol.

Test Cases are specified by a maximum score possible, code to be evaluated, and a function to test whether output is equivalent to what is expected. Test cases will be created by Professors and teaching assistants for the purpose of grading student code.

UNIX [10] is a classification of operating systems derived from the UNIX [10] operating system developed at AT&T's Bell Laboratories in the late 1960s and early 1970s. UNIX [10] systems are known for terse command sets and being relatively simple to develop programs to run on.